Sami Lindqvist

Tasosuunnittelun teoria ja vaiheet

Tasosuunnittelu Source Engine -pelimoottorilla

Metropolia Ammattikorkeakoulu

Medianomi (AMK)

Viestintä

Opinnäytetyö

27.05.2013

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Sami Lindqvist Tasosuunnittelun teoria ja vaiheet 25 sivua + 1 liite 27.05.2013
Tutkinto	Medianomi (AMK)
Koulutusohjelma	Viestintä
Suuntautumisvaihtoehto	3D-animointi ja -visualisointi
Ohjaaja(t)	Lehtori Kristian Simolin

Opinnäytetyön tavoitteena oli selvittää ja ratkaista yleisimmät ongelmat ja haasteet tasosuunnitteluprosessissa käyttäen Valve Corporationin kehittämiä työkaluja Source Engine -pelimoottorille. Tutkielmassa käytiin läpi tärkeimmät periaatteet, toimintatavat ja tuotantovaiheet aina suunnitelmasta julkaisuvalmiiseen pelikenttään asti. Tasosuunnittelun eri vaiheita tarkkailtiin Counter-Strike: Source -pelille luodun Riviera-projektin kautta. Työssä hyödynnettiin Valve Corporationin kattavaa dokumentaatiota työkalujen käytöstä.

Tasosuunnittelu on nykypäivänä hyvin laaja käsite. Sitä voidaan kuvata prosessiksi, jossa luodaan pelaajalle interaktiivinen peliympäristö. Peliympäristön tuotantovaihe pitää sisällään kattavan suunnitelman ideasta luonnokseen, rakenteiden rakentamisen sekä pintamateriaalien ja valaistuksen asettamisen. Teknisen toteutuksen lisäksi suunnittelussa on otettava huomioon pelimekaniikkaan ja pelaajan ohjaukseen liittyviä teorioita.

Lisäksi opinnäytetyössä tutkittiin videopelikehitystä sekä alan osaajien toimintatapoja ja teorioita, millä pyrittiin saattamaan tasosuunnitteluprosessi mahdollisimman kattavasti ja tehokkaasti loppuun asti.

Opinnäytetyössä havaittiin korvaamattomiksi eduiksi kattava suunnitelma ja alan ammattilaisten avaamien teorioiden sekä toimintapojen noudattaminen. Tutkielman lopputuotteena on valmis ja viimeistelty pelikenttä Source Engine -pelimoottorille.

Avainsanat	Tasosuunnittelu, kenttäsuunnittelu, pelisuunnittelu, Source Engine, Valve Hammer Editor, Counter-Strike: Source

Author(s) Title Number of Pages Date	Sami Lindqvist Level design theory and process 25 pages + 1 appendice 27 May 2013
Degree	Bachelor of Culture and Arts
Degree Programme	Media
Specialisation option	3D animation and visualisation
Instructor(s)	Kristian Simolin, Senior Lecturer

This thesis adresses and solves the most common issues and challenges in level design process using tools developed by Valve Corporation for Source Engine. This thesis also analyses the most important principles, procedures and production phases in level design from planning to released product.

Different phases of level design process are examined trough project “Riviera” for a game called Counter-Strike: Source. Study utilizes all inclusive documentation of tools and programs created by Valve Corporation.

Level design is a vast concept nowadays. It can be described as a process where interactive environment is created for a player. Game environment production includes an extensive planning from idea to a design. After building the structures designer needs to add materials and lighting to finish up the product. Game mechanics and theories conserning player control must also taken into account.

This thesis also studies video game development and level design theories by experienced professionals to accomplish polished game environment effeciently. It reveals extensive design and level design theories to be irreplaceable when creating a complete game level for Source Engine.

Keywords	Level design, game design, Source Engine, Valve Hammer Editor, Counter-Strike: Source

Sisällys

Liitteet

Liite 1. Kuvankaappaukset valmiista Riviera-pelikentästä

1 Johdanto

Suomen pelialan hurja kasvuvauhti kiihtyy. Markkinoiden globaalin luonteen ansiosta suurin osa suomalaisesta tuotannosta päätyy ulkomaille. Peliteollisuudesta onkin tullut 2000-luvun aikana merkittävä osa suomalaista kulttuurivientiteollisuutta. Uusien tekijöiden tarve kuitenkin kasvaa myös vuosi vuodelta, mutta ulkomaisten osaajien rekrytointi ei ratkaise ongelmaa, sillä työvoimapulaa on myös muissa valtioissa.

Pelikehityksestä vastaa tuotantotiimi, joka koostuu useista eri jäsenistä rooleineen. Yksi tärkeimmistä osa-alueista onnistuneessa pelielämyksessä on tasosuunnittelu. Tasosuunnittelijan työnkuva on muuttunut vuosien saatossa yksinkertaisten esteiden, vihollisten ja muiden objektien sijoittelusta hyvin laajaksi prosessiksi, jossa ammattitaitoisen suunnittelijan täytyy osata analysoida pelaajan käyttäytymistä luodakseen vakuuttavia pelielämyksiä.

Opinnäytetyöni pääasiallisena tavoitteena on tutkia ja ratkaista yleisimpiä ongelmia ja haasteita tasouunnittelussa käyttäen apuna pelialan ammattilaisten avaamia toimintatapoja ja teorioita. Ensimmäisessä osiossa tarkastellaan Suomen peliteollisuuden tilannetta nykypäivänä sekä tutkitaan pelikehityksen tuotantoprosessia ja tuotantotiimin rakennetta. Toisessa osiossa selvitetään Valve Corporationin Half-Life 2 -tietokonepelille kehittämän 3D-pelimoottorin Source Enginen ominaisuudet. Käytännön osuudessa tarkastellaan mainitun pelimoottorin avulla rakennetun Riviera-tasosuunnitteluprojektin eri vaiheita aina esityöstä julkaisuvalmiiseen pelikenttään asti (kuvio 1).

Kuvio 1: Valmis Riviera-pelikenttä.

2 Videopelikehitys

2.1 Videopelin tuotantoprosessi

Videopelikehitys on prosessi, jossa luodaan videopeli. Tuotannosta vastaa pelikehittäjä, jonka koko voi vaihdella yksittäisestä henkilöstä aina suuriin videopeliteollisuudessa toimiviin yrityksiin asti. Pelikehitystä voidaan pitää normaalina ohjelmiston tuotantoprosessina. Ohjelmistoprojektin hallinnassa viralliset tuontantometodit ja suunnitelmat jätetään usein liian pienelle huomiolle, jolloin kulut ja aikataulu menevät helposti yli budjetin. Suunnittelu on erityisen tärkeää niin yksilö- kuin ryhmäprojekteissakin. Yksi suosituimmista projektinhallinnan viitekehyksistä on niin kutsuttu Scrum-malli, jossa työskennellään toistavasti ja lisäävästi ennustettavuuden optimoimiseksi ja riskien kontrolloimiseksi. Tuotettava peli kehittyy pikkuhiljaa valmiimmaksi tuotteeksi useiden kehitysjaksojen aikana. Näitä kehitysjaksoja kutsutaan sprinteiksi, jotka tavallisesti ovat kestoiltaan 1-4 viikon mittaisia. (Wikipedia 2013a; Wikipedia 2013b; Schwaber & Sutherland 2011.)

Normaalisti pelin kehitystyön rahoittaa kustantaja, mutta niin sanottujen itsenäisten independent-pelitalojen tuotanto etenee usein konseptista prototyyppiin ennen kuin projektin tulevaisuus jätetään kustantajan käsiin. Kustantaja saattaa rahoittaa pelin kehityksen kuukausiksi tai jopa vuosiksi omistaen yksinoikeuden pelin markkinointiin ja levitykseen. Normaalisti pelikehittäjien on työstettävä useampaa tuotetta samanaikaisesti, sillä pelin valmistuminen ja siitä tavoiteltu tuoton saaminen voivat viedä usein jopa vuosia. (Wikipedia 2013a.)

Peliala vaatii jatkuvia innovaatioita, sillä julkaisijat eivät hyödy taloudellisesti loputtomiin kuluneiden ideoiden tai aiempien julkaisujen jatko-osien kierrättämisestä. Uuden pelitalon perustaminen voi olla melko hankalaa, sillä yritys tarvitsee riittävästi alkupääomaa ennen ensimmäisen tuotteen valmistumista ja myyntiä. Tällä hetkellä kuitenkin independent-pelitalot ovat kovassa nosteessa, sillä kasuaali- ja mobiilipelien markkinaosuus on rajussa kasvussa. Etenkin kustannustehokas levitys virtuaalisten sovelluskauppojen kautta on avannut uusia ovia pienemmille yrityksille. Useat independent-pelitalot etenevät pienemmistä tuotannoista suurempiin saavuttaessaan enemmän tuottoa julkaistuista peleistään. Hyviä esimerkkejä tämänkaltaisista yrityksistä ovat suomalaiset Rovio Entertainment ja Supercell. Rovion hittituote ja samalla maailman myydyin mobiilipeli Angry Birds julkaistiin vuonna 2009 saavuttaen

ensimmäisen puolen vuoden aikana 6,5 miljoonan myydyn kopion rajan. (Wikipedia 2013a.) Sittemmin Angry Birds -peliä on ladattu jo huikeat 1,7 miljardia kertaa (Turtola 2013). Vuoden 2012 Pohjoismaiden parhaaksi valitun startup-yrityksen Supercellin Clash of Clans- sekä Hay Day -pelit tekivät Applen kannettavien laitteiden sovelluskaupassa marraskuussa 2012 enemmän liikevaihtoa kuin mikään muu sovelluskehittäjä. Vuonna 2013 Supercellin on kerrottu tuottavan päivässä 1,8 miljoonaa euroa (Lappalainen 2013).

2.2 Tasosuunnittelu ja muut roolit

Peliä kehittää tuotantotiimi, joka koostuu useista eri jäsenistä rooleineen. Joidenkin jäsenien työnkuva saattaa sisältää monia rooleja niiden samankaltaisuuksien vuoksi. Esimerkiksi pelisuunnittelija saattaa usein tehdä myös tasosuunnittelua.

Tasosuunnittelua ei ole helppoa tiivistää yhteen lauseeseen, sillä se on nykyään hyvin laaja käsite. Yleisesti ottaen voidaan kuitenkin sanoa, että tasosuunnittelu on prosessi, jossa luodaan pelaajalle interaktiivinen peliympäristö. 1980-luvun alussa suunnittelu koostui lähinnä erilaisten esteiden, vihollisten ja muiden objektien sijoittelusta peliympäristöön. 1990-luvulla tasosuunnittelu laajeni peleihin kehitetyn kolmiulotteisuuden myötä monimutkaisten sokkeloiden rakentamiseen. 2000-luvulla peliympäristöjen rakentaminen on kehittynyt vielä todella paljon edistyksellisemmäksi ja monimutkaisemmaksi kokonaisuudeksi. Ammattitaitoisessa tasosuunnittelussa ei enää riitä pelkkä kolmiulotteisen tilan hahmotuskyky, vaan se vaatii suunnittelijalta laajaa tietotaitoa suurikokoisten ympäristöjen rakentamisesta kaikkine yksityiskohtineen. Lisäksi pelkkä esineiden ja vihollisten sijoittelu pelitasoon ei riitä, vaan suunnittelijan on osattava skriptata, eli ohjelmoida olioiden käyttäytymistä pelissä toteuttaakseen tavoitellut tapahtumat. Peliympäristön rakentajan täytyy osata analysoida pelaajan käyttäytymistä, jotta ympäristöön voidaan luoda vakuuttavia pelikokemuksia. Tämä vaatii tekijältään vankkaa tuntemusta arkkitehtuurista, taiteesta, ohjelmoinnista, psykologiasta ja graafisesta suunnittelusta. (Johnston 2003.) Tasosuunnittelijaa pidetään usein henkilönä, joka kompositoi koko muun tiimin tuotoksen valmiiksi kokonaisuudeksi. Valmis ympäristö on pelaajalle konkreettisin interaktion kohde ja todella tärkeä osuus pelin menestyksen kannalta.

Tuotantotiimin muut roolit ovat elintärkeitä, jotta tasosuunnittelijan on ylipäätänsä edes mahdollista toteuttaa vakuuttavia peliympäristöjä. Tuotantotiimiä johtaa sisäinen

tuottaja, jonka tehtävänä on muun muassa aikatauluttaa projekti, raportoida työnkulusta ja sijoittaa työntekijöitä työtehtäviin. Pelin suunnittelusta vastaa pelisuunnittelija, joka suunnittelee pelimekaaniset asiat määrittäen pelin rakenteen ja säännöt. Suuressa tuotantoprojektissa voi olla useampia pelisuunnittelijoita, joilla jokaisella on oma suunniteluosa-alueensa. Näitä osa-alueita voivat olla esimerkiksi pelimekaniikat, käyttöliittymäominaisuudet sekä hahmot ja dialogi. Pelin koodia käsittelee ohjelmoija, jonka tehtäviin kuuluu muun muassa implementoida suunnitellut pelimekaaniset elementit peliin. Tuotteen graafisesta ulkonäöstä vastaa artisti, jonka erikoisosaaminen ja työnkuva voi käsittää 2D- tai 3D-grafiikan luomista. 2D-artistit voivat tuottaa konseptitaidetta, peligrafiikkaa tai käyttöliittymägrafiikkaa. 3D-artisti voi luoda esimerkiksi 3D-malleja, 3D-ympäristöjä tai animaatiota. Pelituotannon äänipuolesta vastaa äänisuunnittelija, joka valmistaa ja usein myös sijoittaa äänet pelin sisälle. Laaduntarkkailusta vastaa testaaja, jonka tehtävänä on kontrolloida lopullista laatua varmistaen sen, että peli vastaa pelimekaanisilta ominaisuuksiltaan annettuja ohjeita. (Wikipedia 2013a.)

3 Source Engine -pelimoottori

3.1 Valve Corporation

Source Engine on Valve Corporationin Half-Life 2 -tietokonepelille kehittämä 3D-pelimoottori. Half-Life 2 on niin kutsuttu ensimmäisen persoonan ammunta- eli FPS-peli, jossa pelimaailma esitetään pelihahmon näkökulmasta. Half-Life 2:n lisäksi Source-pelimoottoria on käytetty muissakin menestyneissä peleissä, kuten esimerkiksi Counter-Strike: Sourcessa ja Team Fortress 2:ssa. Poikkeuksena muihin pelimoottoreihin Source Engine on suunniteltu jatkuvasti kehitettäväksi ohjelmistoksi. Tämän mahdollistaa Valve Corporationin luoma jakelu-, moninpeli- sekä viestintäalusta Steam. Päästäkseen käyttämään Valve Corporationin julkaisemia pelejä ja ohjelmistoja on pelaajan asennettava Steam koneellensa. (Wikipedia 2013c; Wikipedia 2013d.)

Source Engine -pelimoottorin pelikentissä käyttämä BSP-tiedostoformaatti pohjautuu id Softwaren vuonna 1996 julkaistun Quaken pelimoottorin vastaavaan formaattiin. BSP on lyhenne sanoista ”Binary Space Partition”. BSP-formaattia on sittemmin käytetty myös muun muassa Quake 2:n ja Half-Life 1:n pelimoottoreissa. BSP-tiedosto sisältää suurimman osan siitä informaatiosta, jota Source Engine -pelimoottori tarvitsee

pelikentän esittämiseen ja pelaamiseen. Tähän sisältyvät esimerkiksi polygonirakenteet, niiden pinnalle piirrettävien pintamateriaalien arvot sekä kaikkien brush-, point- ja prop-entiteettien, eli pelimaailmassa olevien olioiden sijainnit ja ominaisuudet. (Valve Developer Community 2013a.)

3.2 Counter-Strike: Source

Counter-Strike: Source on Source Engine -pelimoottorille käännetty versio Counter-Strike -modifikaatiosta (kuvio 2). Counter-Strike: Source on internetissä pelattava joukkuepohjainen ja nopeatempoinen FPS-tietokonepeli, jossa terroristeista ja erikoisjoukoista koostuvat joukkueet tavoittelevat erävoittoja kahdella pääasiallisella tavalla: suorittamalla määrätty tehtävä tai eliminoimalla vastapuolen pelaajat. Virallisissa pelimuodoissa terroristien tavoitteena on pommien viritys ja panttivankien hallussapitäminen. Erikoisjoukkojen tehtävänä on vastaavasti purkaa asetettu pommi tai pelastaa panttivangit. Kentän koosta, pelaajamäärästä ja pelimuodosta riippuen erät kestävät noin yhdestä neljään minuuttia. (Wikipedia 2013e.)

Kuvio 2: Counter-Strike: Source.

Valve Corporationin ja Turtle Rock Studiosin kehittämä Counter-Strike: Source on tunnettu sen laajasta ja omistautuneesta kannattajajoukosta, joka on tuottanut pelille runsaasti lisäsisältöä. Lisäksi pelin ammattitaitoisia joukkueita varten järjestetään useita kansainvälisiä turnauksia, joissa palkintorahat yltävät kymmeniin tuhansiin euroihin. (Wikipedia 2013e.)

3.3 Source Engine -pelimoottorin ominaisuudet

Source Engine tarjoaa käyttäjälleen nopean, luotettavan ja joustavan alustan vakuuttavien peliympäristöjen tekemiseen. Shader-pohjainen renderöintiteknologia mahdollistaa hankalimpienkin ympäristöjen mallintamisen nopeasti ja tehokkaasti. Source hyödyntää myös kehittynyttä prosessoriteknologiaa, kuten esimerkiksi moniydinprosessoreita ja SIMD-teknologiaa. Lisäksi se kykenee hyödyntämään uusimmat DirectX-ohjelmointirajapintaa käyttävät grafiikkaprosessorio minaisuudet. (Valve Developer Community 2012a.)

Edistyksellinen Shader-teknologia tukee Microsoftin kehittämää High Level Shader Language -kieltä (HLSL) käytettäväksi Direct3D-ohjelmointirajapinnan kanssa. Lisäksi kattava Shader-kirjasto antaa käyttäjälleen mahdollisuuden luoda monipuolisia valaisutekniikoita Half-Life 2:n realistisesta tyylistä aina Team Fortress 2:n kaltaiseen piirrosmaiseen tyyliin. Tehokkaan suorituskyvyn takaamiseksi Source Engine käyttää 3D-malleissaan Level of Detail -teknologiaa (LOD), joka vähentää objektien yksityiskohtien määrää katsojan siirtyessä kauemmaksi kohteesta. (Valve Developer Community 2012a.)

Mukaansatempaavien ympäristöjen geometria valaistaan joko valokarttoihin sisällytetyn radiositeettivalaistuksen tai verteksivalaistuksen kautta. Valokartat toimivat hyvin töyssyjen, naarmujen ja muiden pinnan muotoja määrittelevien niin sanottujen Bump-karttojen kanssa luoden objekteihin tarkemman valon ja varjon. Lisäksi kaikki valaistusinformaatio lasketaan valaistusta tasoittavassa High Dynamic Range (HDR) -valaistustilassa luonnollisen valaistuksen saavuttamiseksi. (Valve Developer Community 2012a.)

Dynaamiset objektit ja hahmot voivat vastaanottaa ennalta laskettua heijastuvaa valoa ympärillä olevasta maailmasta saaden objektit sopimaan ympäristöönsä täydellisesti. Ilmiötä tehostaa dynaamisten objektien ja hahmojen kyky projektoida korkealaatuisia reaaliaikaisia varjoja ympäröivään geometriaan. (Valve Developer Community 2012a.)

Source Enginellä voidaan luoda runsaasti erilaisia erikoisefektejä. Sumun, sateen ja muiden vastaavien efektien lisäksi voidaan luoda partikkelisysteemejä, jotka tuottavat muun muassa uskottavaa tulta, räjähdyksiä tai lunta. Partikkelieditorin avulla tuotetut efektit voidaan nähdä välittömästi peliympäristössä. Lisää realistisuutta elämykseen tuo

ympäristöä heijastava ja valoa taittava vesi sekä nopean kameran liikkeen generoima liike-epäterävyys. (Valve Developer Community 2012a.)

Source Engine -pelimoottorin materiaalisysteemi sisältää kokoelman materiaaleja, jotka määrittävät, mistä aineesta objektit on tehty ja mitä pintamateriaalia ne käyttävät. Nämä määritelmät kertovat muun muassa miten objektit käyttäytyvät hajotessaan, mitä ääntä ne päästävät osumahetkellä tai raahautuessaan toista materiaalia vasten sekä mitä objektien massat ja kelluvuudet ovat. Muiden hienovaraisempien materiaalikohtaisten ominaisuuksien lisäksi Source Engine tarjoaa mahdollisuuden koko pelikentän kattavalle värikorjailuille, joka mahdollistaa esimerkiksi kontrastin ja saturaation muokkaamisen interaktiivisesti pelin sisällä. (Valve Developer Community 2012a.)

Yhteensopivuus suosituimpien grafiikka- ja 3D-mallinnusohjelmien kanssa mahdollistaa realististen tai tyyliteltyjen hahmojen, aseiden, ajoneuvojen ja muiden objektien mallintamisen Source Enginen ympäristöön. Runsas työkalujen määrä auttaa tehokkaasti riggaamaan, animoimaan sekä määrittämään fyysisiä interaktioita hahmojen ja muiden objektien välille. Valve Corporationin kehittyneet animaatiotyökalut tarjoavat luurankoihin perustuvan animaatiosysteemin, jolla voidaan luoda hahmomalleille sulavia ja monimutkaisia vartalon liikkeitä. Työkalut tarjoavat myös edistykselliset animointimahdollisuudet hahmomallien kasvoille. Uskottavien hahmojen luomisessa auttaa simuloitu lihaksisto, jonka avulla voidaan projektoida hahmojen tunnetiloja, puhetta ja vartalon elekieltä. Ympyrän muotoisten ja valoa heijastavien silmien ansiosta katse voidaan keskittää tarkemmin esimerkiksi pelaajaan tai objekteihin. (Valve Developer Community 2012a.)

Source Enginen suorituskyvyltään tehokaalla fysiikkasysteemillä voidaan luoda reagoivia ja muuttuvia ympäristöjä, joissa tekoälyhahmot ovat interaktiossa fyysisesti simuloitujen objektien kanssa. Näitä fysiikkaominaisuuksia kontrolloidaan tasosuunnittelun kautta, jolloin voidaan rakentaa muun muassa erilaisia koneita, ajoneuvoja, köysiä, kaapeleita tai muotoaan muuttavia objekteja. (Valve Developer Community 2012a.)

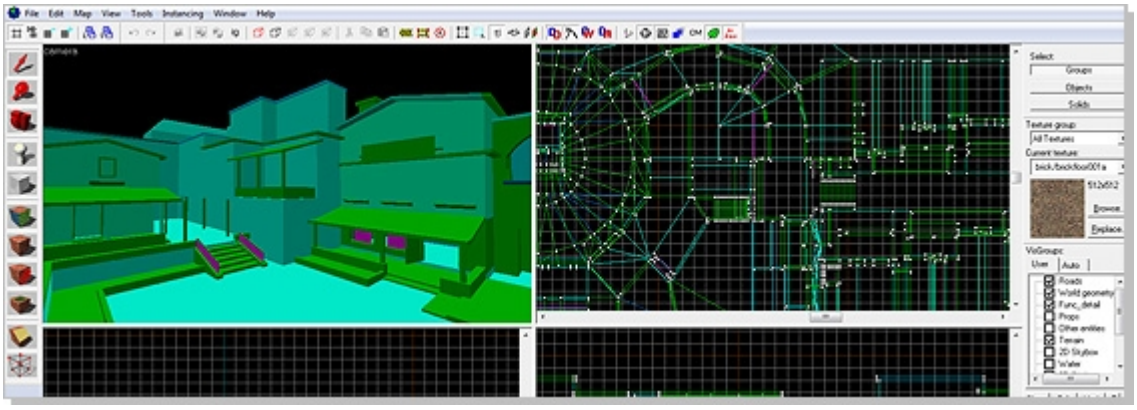
4 Tasosuunnittelun vaiheet

4.1 Esityö

4.1.1 Työkalut

Tasosuunnittelussa varsinainen tekninen toteutus tapahtuu tarkoitukseen sopivilla suunnittelutyökaluilla. Näitä työkaluja voivat olla muun muassa 3D-mallinnusohjelmat 3D Studio Max ja Maya. Usein kuitenkin pelin mukana tulevat kehittäjän luomat, spesifimmät suunnittelutyökalut.

Valve Corporation tarjoaa Source Engine -pelimoottorille todella kattavat välineet pelinkehitykseen Steam-ohjelmiston kautta. Source Software Developement Kit eli Source SDK sisältää oleellisimmat työkalut Half-Life 2 -modifikaatioiden luomiseen. Tasosuunnittelijalle tärkein on Valve Hammer Editor (kuvio 3), jolla luodaan pääsääntöisesti kaikki pohjarakenteet ympäristöille. (Wikipedia 2013f.)



Kuvio 3: Valve Hammer Editor.

4.1.2 Idea

Ensimmäinen vaihe tasosuunnittelussa on saada idea siitä, mitä ryhtyy rakentamaan. Suunnittelijan on helpompaa luoda uskottava ympäristö pelaajille, mikäli kentän taustalla on vakuuttavasti pohjustettu taustakertomus sijainnista ja tapahtumasta. (World Of Level Design 2011a; World Of Level Design 2011b.) Pelaajan on tasoa pelatessaan tärkeää tuntea oikeasti liikkuvansa luodussa maailmassa. Usein tämä tuntuu olevan kompastuskivi. Peliympäristöt ovat liian geneerisiä eivätkä esitä lainkaan sitä maailmaa, johon kentän pitäisi sijoittua.

Riviera-kenttäprojektin taustalla olevaa ideaa raikkaasta välimerellisestä ympäristöstä pelimaailmassa oli helppo lähestyä. Kyseisessä ympäristössä oli luonnollista leikitellä runsailla väreillä, jotka uniikin visuaalisen ilmeen lisäksi saavat pelaajat mahdollisesti myös viihtymään moninpelikentän parissa pidempään.

4.1.3 Referenssit ja luonnostelu

Referenssikuvat ovat lähes välttämättömiä niin tasosuunnittelussa kuin pintamateriaalien ja 3D-mallienkin luomisessa (World Of Level Design 2011a). Mallikuvista pystyy helposti toteamaan muun muassa yksityiskohtien tärkeyden realismiin pyrkiessä.

Riviera-projektia varten keräsin satoja kuvia erilaisiin käyttötarkoituksiin. Näitä käyttötarkoituksia olivat muun muassa referenssit oikean tyylin luomiseen kentän arkkitehtuurissa sekä opastaminen kentän värimaailmaan ja valaistukseen. Lisäksi esimerkiksi pintamateriaalien ja 3D-mallien tekeminen vaati omat referenssinsä.

Luonnosteluvaihe on yksi tärkeimmistä, ellei tärkein tasosuunnittelun osa-alueista. Se on koko prosessin kestoa ajatellen todella nopea työvaihe, joka voi hyvin tehtynä vauhdittaa työntekoa moninkertaisesti. Luonnosten tarkoituksena on helpottaa tekijäänsä myöhemmissä tuotannon vaiheissa. Luonnostelua voi tehdä esimerkiksi piirtämällä paperille raakaversion pelikentän pohjapiirustuksesta eli navigaatiorakenteesta. Myös yksityiskohtaisempia hahmotelmia kentän eri alueista voi tehdä. Tärkeintä on se, että kentän pääasialliset ideat ja tavoitteet tulevat esille. Luonnostelun tärkeys käy ilmi myös projektin loppuunviemisessä. Ajatustyön liiallinen jättäminen itse tuotantovaiheeseen voi olla erittäin tuhoisaa tuotteen valmistumista ajatellen. Mielenkiinnon pysyminen projektissa on paljon helpompaa, kun tietää, mitä seuraava työvaihe pitää sisällään. (World Of Level Design 2011b.)

Riviera-kentän pohjapiirustusta luonnostellessa olennaisin asia oli kartoittaa tason rakenne niin, että se toimi pelillisesti mahdollisimman hyvin tiimipohjaisessa, nopeatempoisessa tulitaistelussa. Tärkeää oli myös ottaa huomioon Source Engine -pelimoottorin tekniset rajoitukset, jotka toivat omat haasteensa esimerkiksi laajojen ja avarien ulkoilma-alueiden luomiseen. Näitä asioita silmällä pitäen suunnittelin näyttäviä ja helposti muistettavia alueita, jotka sijoitin paperille alustavasti. Kyseisiä lokaatioita olivat muun muassa terroristitiimin aloituspaikka, jonka oli tarkoitus olla pienehkö

satama-alue meren rannalla sekä pommipaikka B, joka esitti lähetystörakennusta (kuvio 4). Pohjapiirustuksen luonnostelussa pyrin lisäksi ottamaan huomioon tason sijoittumisen suurehkolle rinteelle, jolloin horisontissa näkyi jatkuvasti rakennuksia, palmuja ja muita yksityiskohtia. Tämä toi sekä kentän visuaaliseen ilmeeseen että pelimekaanisiin seikkoihin huomattavan paljon lisää syvyyttä.

Kuvio 4: Riviera-pelikentän lopullinen navigaatiorakenne.

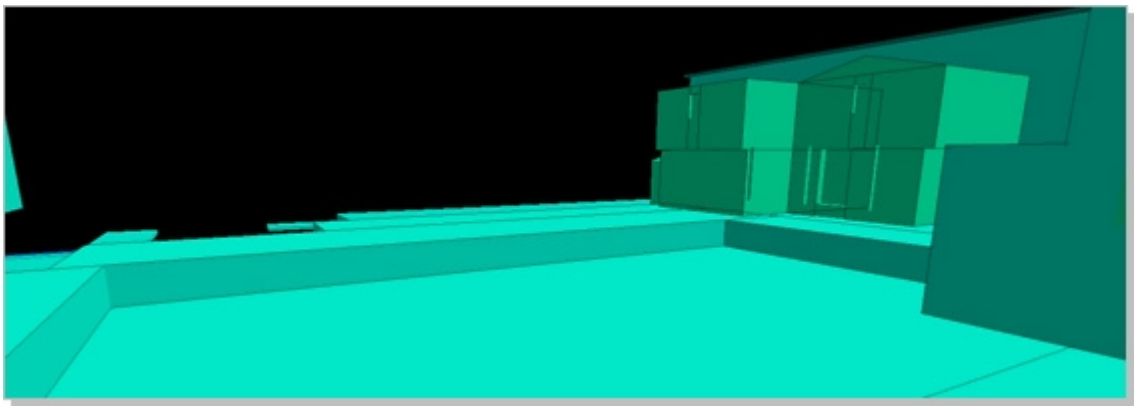
4.2 Perusrakenteet

Konkreettinen pelikentän rakennus alkaa luomalla perusrakenteet. Perusrakenteilla tarkoitetaan lähinnä primitiivisiä muotoja, joilla rakennetaan esimerkiksi rakennuksien muodot, tasanteet ja muu maasto. Perusrakenteilla määritellään hyvin pitkälti kentän lopullinen navigaatiorakenne, jolloin aiemmin mainitun luonnostelun tärkeys tulee hyvin ilmi.

Kaikki Valve Hammer Editor -ohjelmassa rakennetut pinnat tehdään BSP-teknologialla. Näitä Block Tool -työkalulla luotuja pintoja kutsutaan usein brusheiksi, joilla kaikki mainitut perusrakenteet luodaan. Normaalit brushit on tarkoitettu pääsääntöisesti hyvin yksinkertaisten muotojen tekemiseen, eikä niitä voida suositella käytettäväksi monimutkaisissa muodoissa. Väärin käytettyinä nämä monimutkaiset muodot, kuten

esimerkiksi pyöreä pallo saattavat heikentää dramaattisesti pelikentän suorituskykyä. (Valve Developer Community 2012b.)

Riviera-tasossa aloitin perusrakenteiden tekemisen terroristitiimin aloituspaikasta rakentamalla koko alueelle tasanteet, joilla pelaajat kulkevat. Tämän jälkeen rakensin tasanteiden päälle hyvin yksinkertaiset hahmotelmat asuinrakennuksista, jotka korvaisin myöhemmin yksityiskohtaisemmilla versioilla (kuvio 5). Aloituspaikan hahmotelman jälkeen etenin systemaattisesti kentässä eteenpäin toistamalla samat toimenpiteet A- ja B-pomminpaikoista sekä näiden välimaastoista. Lopuksi hahmottelin erikoisjoukkojen aloituspaikan, joka sijaitsi suhteessa terroristitiimiin kentän vastakkaisella puolella.



Kuvio 5: Riviera-pelikentän perusrakenteet.

Ennen siirtymistä tarkempiin yksityiskohtiin, rakensin tasoon vielä niin sanotun 2D-skybox-elementin, jonka tehtävänä on piirtää kentän ympärille renderöity 2D-kuva taivaasta. Samalla otin käyttöön VisGroups-ominaisuuden, jolla voidaan näyttää ja piilottaa eri tasoille määriteltyjä objektiryhmiä. Asetin perusrakenteille sekä 2D-skyboxille omat VisGroup-ryhmänsä, jotta pystyin halutessani piilottamaan taivaan editorissa. VisGroups-ominaisuus helpottaa työstämään pelikenttää tuotannon myöhemmässä vaiheessa objektien määrän noustessa kymmeniin tuhansiin.

4.3 Rakenteelliset yksityiskohdat

4.3.1 Yksityiskohdat ja suorituskyky

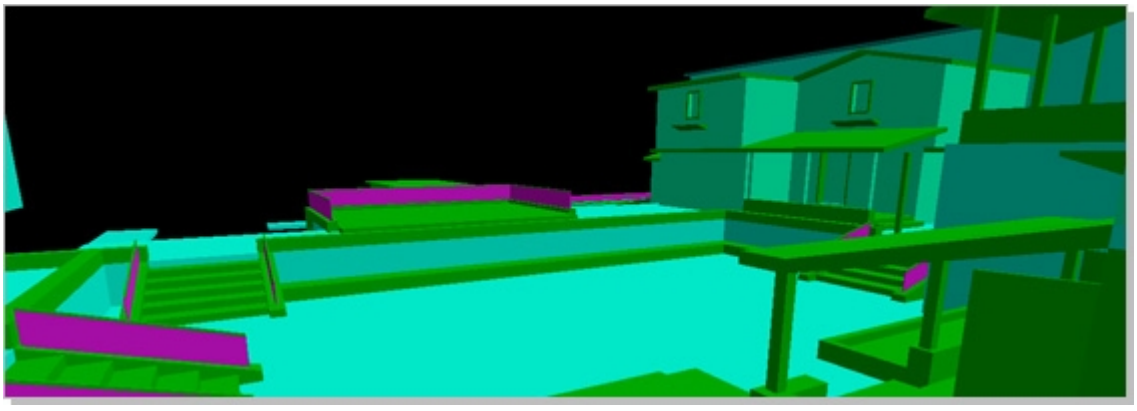
Pelimaailmassa suunnittelijat joutuvat jatkuvasti tekemään kompromisseja ulkonäön suhteen, sillä suurempi määrä yksityiskohtia reaaliaikaisessa 3D-ympäristössä

hidastaa tietokoneen suorituskykyä. Tämän vuoksi on tärkeää sijoittaa yksityiskohtia alueille, joissa pelaaja kulkee sekä niiden välitön läheisyys. Epäolennaisemmat asiat, kuten esimerkiksi kauempana horisontissa näkyvä maasto tai rakennukset voidaan hyvin jättää pelkistetyimmiksi elementeiksi.

4.3.2 Yksityiskohtien lisääminen

Entiteetit ovat Source Engine -pelimoottorissa objekteja, joilla on joitain erityisiä, normaaleista brusheista poikkeavia ominaisuuksia. Oleellisimpia entiteettejä ovat brush- ja point-entiteetit. Brush-entiteetit ovat entiteettejä, jotka on sidottu yhteen tai useampaan brushiin. Näiden objektien muoto ja ulkonäkö määrittyvät niihin sidottujen brushien mukaan. Point-entiteetit ovat ympäristöön asetettavia nimensä kaltaisia pisteitä, joilla ei itsessään ole tiettyä vaikutusaluetta. Point-entiteetit ovat yleisimmin esimerkiksi tekoälyä tai valaistusta määritteleviä objekteja, mutta niitä käytetään myös prop-entiteetteinä, joilla tuodaan pelikenttään erilaisia staattisia, dynaamisia tai fyysisiä 3D-malleja. (Valve Developer Community 2012b.)

Riviera-projektissa yksityiskohtien lisäämisen aloitin pääsääntöisesti brush-entiteeteistä. BSP-pohjaiset visuaaliset yksityiskohdat loin func_detail-nimisillä brush-entiteeteillä, joita kutsutaan usein yksinkertaisemmin detail-brusheiksi. Nämä detail-brushit eivät vaikuta kentän laskentaprosessiin eivätkä myöskään pilko muita normaaleja brusheja osiin. Käytännössä kaikkien objektiten, jotka eivät ole näkyvyyttä määritteleviä perusrakenteita pitäisi olla sidottuja func_detail-entiteetteihin. (Valve Developer Community 2012b.) Näillä entiteeteillä lisäsin kenttään esimerkiksi asuintalojen parvien ja kattotasanteiden kaltaisia yksityiskohtia sekä portaita (kuvio 6).



Kuvio 6: Riviera-pelikenttään lisätyt detail-brushit.

Brush-pohjaisten entiteettien jälkeen siirryin koristelemaan tasoa point-entiteetein aseteltavilla 3D-malleilla. 3D-mallien käyttö peliympäristöissä on yleistynyt tasaisesti viimeisen kymmenen vuoden aikana. Nykyään suurin osa peliympäristöistä rakentuu pääosin 3D-mallien varaan. Source Engine -pelimoottorissa kuitenkin BSP-tekniikat ovat vielä jokseenkin suuremmassa roolissa, joten 3D-malleja käytetään lähinnä tuomaan kenttään viimeiset rakenteelliset yksityiskohdat.

Pelissä käytettävää 3D-mallia lähdetään luomaan erillisessä 3D-mallinnusohjelmassa. Näitä malinnusohjelmia ovat muun muassa Softimage XSI, 3D Studio Max ja Maya. Työnkulku etenee ideasta luonnokseen ja pohjapiirustukseen, jonka jälkeen mallinnetaan objektin geometria. Tämän jälkeen 3D-mallille määritetään pinnan sulavat muodot ja terävät reunat, jotta malli voidaan valaista oikein. Lopuksi määritellään vielä UV-koordinaatit, joiden avulla pintamateriaali voidaan heijastaa objektin pinnalle oikein. Objektille täytyy myös usein luoda kollisiomalli, joka määrittää 3D-mallin rakenteellisen pinnan pelimoottorissa.

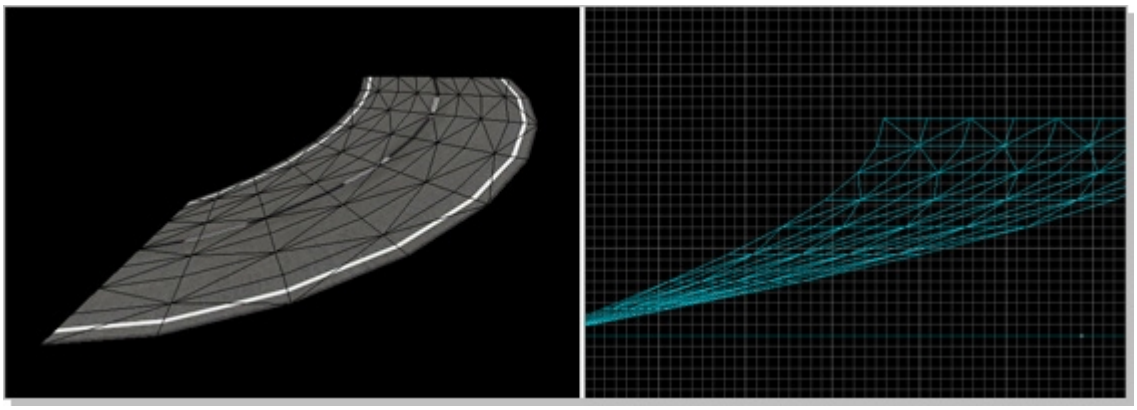
Staattiset 3D-mallit lisätään Valve Hammer Editorissa pääosin prop_static-entiteeteillä ja fyysisiä ominaisuuksia omaavat objektit prop_physics- sekä prop_physics_multiplayer-entiteeteillä (Valve Developer Community 2012b). Riviera-kentässä käytin yhteensä satoja 3D-malleja herättämään ympäristön henkiin (kuvio 7). Suurin osa näistä objekteista oli pelin mukana tulleita yleishyödyllisiä esineitä. Käytössä oli myös satunnaisia, vapaasti käytettäviä kolmannen osapuolen eli ulkoisen kehittäjän valmistamia 3D-malleja. Nämä kolmannen osapuolen luomat objektit olivat todella tärkeitä tuomaan kentälle uniikin ilmeen, sillä pelin mukana tulleet 3D-mallit ovat pelaajille jo liiankin tuttuja.

Lähes kaikki kenttään sijoitetut 3D-mallit olivat staattisia prop-malleja sisältäen muun muassa puita, hylättyjä ajoneuvoja ja yksityiskohtia rakennuksiin. Lisäksi mukana oli fyysisiä ominaisuuksia omaavia prop-malleja, kuten esimerkiksi hajoavia kukkaruukkuja, lasipulloja ja mainoskylttejä. Yritin kuitenkin pitää fyysisiä ominaisuuksia sisältävien objektien määrän mahdollisimman pienenä, jotta laajoja ja avaria alueita sisältävä kenttä säilyttäisi hyvän suorituskykynsä hitaammallakin tietokonekokoonpanolla.

Kuvio 7: Riviera-pelikenttään lisätyt 3D-mallit.

Hankalampia muotoja varten Valve Corporation on kehittänyt Source-pelimoottorilleen brush-pohjaiset polygonimesheiksi konvertoidut displacement-objektit. Näitä displacement-brusheja voidaan veistää ja vääntää monimutkaisemmiksi muodoiksi, mikä mahdollistaa muun muassa paljon korkeuseroja omaavan maaston tai kallion rakentamisen. (Valve Developer Community 2012c.)

Yksi suurimmista haasteista teknisesti Riviera-tasossa olivat kaarevat auto- ja kävelytiet (kuvio 8). Kaarevien teiden rakentaminen detail-brush-tekniikalla olisi ollut helpoin vaihtoehto, mutta displacement-brushien käytön keveys suorituskyvyn suhteen yhdistettynä sulavampaan lopputulokseen oli kuitenkin vaivan arvoinen työtehtävä. Valve Hammer Editorin displacement-työkalut olivat hieman rajalliset, minkä vuoksi tarvitsin teiden luomiseen kolmannen osapuolen ohjelmiston nimeltään Twister, jonka avulla pystyin generoimaan helposti ja nopeasti muotoja, jotka normaalisti olisivat hyvin vaikeita, ellei jopa mahdottomia luoda Valve Hammer Editorissa (Twister Homepage 2009).



Kuvio 8: Displacement-brush.

4.4 Pintamateriaalit

4.4.1 Pintamateriaalit ja värioppi

Antaakseen peliympäristön objekteille konkreettisen materiaalin on tasosuunnittelijan määriteltävä erikseen jokaiselle pinnalle pintamateriaali koordinaatteineen. Pintamateriaali sisältää erilaisten käyttätymistä määrittelevien ominaisuuksien lisäksi informaation piirrettävistä tekstuureista eli kaksiulotteisista kuvista, jotka heijastetaan objektin pinnalle. Suurin osa tekstuureista on niin kutsuttuja diffuse-karttoja, jotka määrittävät materiaalin värit. Muita tekstuureita ovat muun muassa bump- ja specular-kartat, jotka määrittävät enemmän pinnan käyttäytymistä esimerkiksi valon ja varjon kanssa, kuin itse pinnan väri-informaatiota. (Valve Developer Community 2011b.)

Värioppi käsittelee värejä ja niiden käyttöä. Tasosuunnittelussa väriopin avulla voidaan ohjata pelaajaa siinä missä äänillä, muodoilla ja valollakin. Väriopin keskeisimpiä käsitteitä ovat yleisen väriharmonian lisäksi päävärit keltainen, punainen ja sininen sekä toisensa poissulkevat vastavärit keltainen-violetti, punainen-vihreä ja sininen-oranssi. Goethen väriympyrästä (kuvio 9) on nähtävissä, kuinka rinnakkain olevat värit sulautuvat hyvin toisiinsa samalla, kun vastakkaiset värit luovat vakuuttavimmat kontrastit. (Wikipedia 2013g.) Tasosuunnittelussa näiden perusperiaatteiden avulla voidaan luoda ympäristöön tehokkaita huomiopisteitä käyttäen oikeanlaisia tekstuureita. Pintamateriaaleja asettaessa kannattaa kuitenkin muistaa, että sen tehtävä on tukea väreillään valaistusta. Hyvällä teksturoinnilla voidaan pelastaa tai pilata valaistus (Hourences 2006).

Kuvio 9: Goethen väriympyrä. (Wikipedia 2013g.)

4.4.2 Pintamateriaalien asettaminen

Valve Hammer Editorissa pintojen materiaalien koordinaatit, skaalaukset ja rotaatiot määritellään Texture Application -työkalulla. Työkalun avulla voidaan määrittää eri materiaali objektin jokaiselle pinnalle yksilöllisine arvoineen. (Valve Developer Community 2013b.)

Riviera-tason teksturoinnin aloitin itsetehdyillä pintamateriaaleilla (kuvio 10). Pelin mukana tulleiden pintamateriaalien värikylläisyys ei riittänyt halutun tunnelman luomiseksi, joten omat tekstuurit olivat välttämättömiä erittäin värikkään ja raikkaan ympäristön aikaansaamiseksi. Erityisesti taivas eli 2D-skybox, meri sekä pelaajaa lähinnä olevat asuinrakennukset vaativat melko räikeät pintamateriaalit. Pyrin myös löytämään kolmannen osapuolen tuottamia tekstuuripaketteja värittääkseni puuttuvat pinnat. Pelin omia materiaaleja käytin vain välttämättömissä ja usein varsin huomaamattomissa kohteissa.

Kuvio 10: Itsetehtyjä tekstuureita Riviera-pelikenttään.

4.5 Valaistus

4.5.1 Valaistus peliympäristöissä

Valaistus on yksi merkittävimmistä vaiheista tasoa rakentaessa. Toimivan valaistuksen rakentaminen jätetään usein liian vähälle huomiolle eikä suunnittelijoilla ole tarpeeksi kiinnostusta panostaa siihen. Valolla on voima muuttaa ympäristö pelottavaksi, mukavaksi, kylmäksi tai kuumaksi. Lisäksi valaistuksella voidaan tehostaa objektien kolmiulotteisuutta sekä luoda toimiva kompositio ja balanssi, joilla johdatellaan pelaajan katsetta ja liikettä tarkoituksenmukaisiin kohteisiin. (Hourences 2006; Jenssen 2012.)

Ensimmäinen ja yksi tärkeimmistä valaistuksen osa-alueista on valonlähde. Valoa ei tule tyhjästä, joten valolla on aina oltava lähde. Valonlähteitä voivat olla perinteisten lamppujen lisäksi muun muassa aurinko, kuu, tuli tai veden heijastava pinta. Tärkeää on varmistaa, että valon voimakkuus ja väri ovat tasapainossa valon lähteen kanssa; yksi pöytälamppu ei kykene valaisemaan kokonaista tehdashallia. Riittävää valon vaikutelmaa voidaan tehostaa muun muassa erilaisilla erikoisefekteillä, kuten hehkuilla tai linssiheijastuksilla. (Hourences 2006.)

Valon värittömyys on usein toinen kompastuskivi, sillä jokaisella valolla on lähes aina tylsän valkoisen sijasta enemmän tai vähemmän värisävyä. Esimerkiksi vanha ja kulunut pöytälamppu voi loistaa kellertävää valoa tai valo voi kulkea sinertävän lasin läpi luoden kylmempää sävyä ympäristöön. Yleisesti voidaan sanoa, että valonlähde lähettää sähkömagneettista säteilyä sitä enemmän, mitä korkeampi sen lämpötila on. (Hourences 2006.)

Valonsäteet voivat myös kimmota erilaisilta pinnoilta eteenpäin värittäen maailmaa pintamateriaalin mukaan. Nykypäivän pelimoottorit eivät vielä täysin pysty tarjoamaan tämän kaltaista radiositeettivalaistusta, joten suunnittelija joutuu usein ottamaan huomioon vallitsevan ympäristön oikeanlaisen valon voimakkuuden ja värin valitsemiseksi. (Hourences 2006.)

4.5.2 Valojen asettaminen

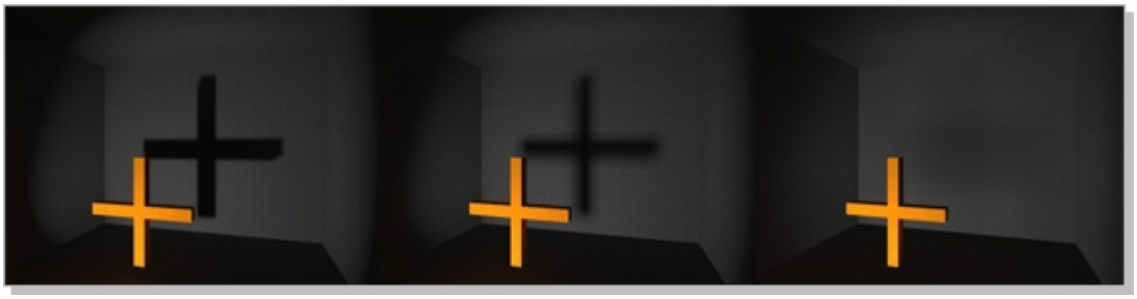
Source-pelimoottorin taso vaatii valaistuksen. Ilman valaistusta ympäristöstä tulee oletusarvoisesti täysin valaistu, jolloin pinnoille ei luoda valoa ja varjoa. Täysin valaistu kenttä ilman valoa ja varjoa ei ole käytännössä koskaan ideaali tilanne peliympäristössä. (Valve Developer Community 2013c.)

Riviera-tason valaisemisen aloitin lisäämällä auringonvalon light_environment-entiteetillä, joka simuloi yhdensuuntaisen valon ulkotiloihin käyttäen valonlähteenä kenttää ympäröivää taivasta. Auringonvaloon voidaan määrittää esimerkiksi valon suunta, kallistuskulma, voimakkuus ja väri (Valve Developer Community 2013c). Riviera-kenttä sijoittui ajallisesti lähes pilvettömään keskipäivään, joten pyrin asettamaan valon kulman melko pystysuoraksi ja asettamaan valon värin hieman kellertäväksi.

Sisätilat, joihin auringonvalo oli riittämätön valaisin pääosin light-entiteeteillä. Light-entiteetti on staattinen pistevalo, jota voidaan käyttää esimerkiksi lampuissa ja kynttilöissä. Tämän lisäksi sisätiloja valaistaessa käytin useita light_spot-entiteettejä, jotka nimensä mukaisesti ovat kohdistettuja spottivaloja. Source tarjoaa lisäksi dynaamisen valonlähteen, joka lisätään light_dynamic-entiteetillä. (Valve Developer Community 2013c.)

4.5.3 Valokartat ja tasaisuusarvot

Valokartat pitävät sisällään jokaisen pinnan valaistusinformaation. Valokartan numeerinen arvo määrittää tekstuurin pikselien määrän jokaista valon pikseliä kohden. Pienempi arvo luo pinnalle terävämmän ja tarkemman valaistuksen, kun taas suurempi arvo luo epätarkan ja sumean valaistuksen (kuvio 11). (Valve Developer Community 2013c.)



Kuvio 11: Valokartta-arvot 4, 16 ja 64. (Valve Developer Community 2013c.)

Tasaisuusarvoja eli Smoothing Group -asetuksia käytetään määrittelemään valaistuksen pehmeys eri pintojen välillä. Pääsääntöisesti tasaisuusarvoja käytetään yhtenäisissä pyöreissä muodoissa, jotta vältetään terävät varjon ja valon siirtymäkohdat. Klassinen esimerkki tasaisuusarvojen tarpeellisesta käytöstä on sylinterin muotoinen objekti. Sylinteri kadottaa pehmeän muotonsa mikäli arvoja ei määritellä, sillä jokainen sylinterin sivu valaistaan erikseen (kuvio 12). (Valve Developer Community 2013c.)

Tasaisuusarvot ja tarkat valokartat eivät vaikuta varsinaisesti valmiin pelikentän suorituskykyyn, mutta ne voivat nostaa kentän kääntämisprosessiin käytettävää aikaa dramaattisesti. Valve Hammer Editorissa tasaisuusarvot ja valokartat määritellään pintamateriaalien tavoin Texture Application -työkalulla. (Valve Developer Community

2013c.) Riviera-tasossa pyrin asettamaan näkyvimmille alueille mahdollisimman terävät varjot suoran auringonvalon vuoksi.

Kuvio 12: Vasemmalla ei Smoothing Group -määrityksiä. Keskellä Smoothing Group -määritykset suurella valokartta-arvolla. Oikealla Smoothing Group -määritykset pienellä valokartta-arvolla. (Valve Developer Community 2013c.)

4.5.4 HDR-valaistus

Tavanomaisen Low Dynamic Range (LDR) -valaistusinformaation lisäksi Source Engine tarjoaa tuen monipuolisemmalle HDR-valaistustilalle (kuvio 13). LDR-valaistus vaatii suunnittelijalta usein enemmän manuaalista pistevalojen asettelua, jolla korjataan valon heikkoa kantautumista pelikentän pimeimpiin alueisiin. HDR-valaistustila sen sijaan kykenee tasoittamaan valaistusta pimeämmillä alueilla automaattisella valotuksella. Haittapuolena HDR-valaistuksessa on BSP-tiedoston kasvava pakettikoko, sillä LDR- ja HDR -valaistustilat vaativat erilliset valaistusinformaatiot eli valokartat. (Valve Developer Community 2013c.)

Kuvio 13: Riviera-pelikenttä LDR- ja HDR-valaistuksessa.

4.6 Viimeistely

4.6.1 Erikoisefektit

Heijastavat pinnat vaativat ennalta lasketun renderöinnin ympäristöstä, jotta heijastukset pystytään piirtämään oikein. Suunnittelija sijoittaa pelikenttään env_cubemap-entiteettejä, joista renderöidyt kuvat tallennetaan niin kutsuttuina cubemap-tekstuureina BSP-tiedostoon. Jokainen ympäristöön sijoitettu heijastava pintamateriaali hakee heijastettavan informaation lähimmästä env_cubemap-entiteetistä, ellei suunnittelija toisin määrittele. LDR- ja HDR-tilat vaativat erilliset cubemap-tekstuurit, jotka kasvattavat BSP-tiedoston pakettikokoa. (Valve Developer Community 2013d.)

Tavanomaisen 2D-skyboxin lisäksi rakensin Riviera-kenttään täysin erillisen 3D-skyboxin. 3D-skybox on pelikentästä erilleen rakennettu ja pieneksi skaalattu alue, jonka tasosuunnittelija sijoittaa kenttäeditorissa pelialueen ulkopuolelle pelaajien ulottumattomiin. Kun pelikenttä ladataan peliin piirretään tämä pienikokoinen 3D-skybox suuremmassa mittakaavassa pelialueen ja 2D-skyboxin välille, jolloin voidaan saavuttaa hyvin voimakas tunne todella isosta ympäristöstä pelialueen ympärillä (kuvio 14). Riviera-tasossa 3D-skybox oli elintärkeä osa oikeanlaisen tunnelman ja syvyysvaikutelman aikaansaamiseksi. (Valve Developer Community 2012d.)

Kuvio 14: Riviera-pelikentän 3D-skybox-elementti.

Valaistuksen näyttävyyttä tehostin env_sprite-entiteeteillä, joiden tehtävänä oli luoda hohto-efekti valon ympärille. Vastaavanlaisen efektin lisäsin myös auringonvalolle env_sun-entiteetillä, jolloin valo häikäisi näkökentän pelaajan kääntäessä katseensa aurinkoa vasten. Viimeisimpiä visuaalisia yksityiskohtia olivat muun muassa

func_dustmotes-entiteeteillä lisätyt pölypartikkelit, env_steam-entiteetillä luotu moottorin lämpöväreily, erilaiset sähkökaapelit sekä info_overlay-entiteeteillä luodut graffitit ja muut pistemäisesti sijoitetut tekstuurit.

Loppusilauksen kentän visuaaliseen ilmeeseen tein Source Enginen tarjoamalla Color correction -ominaisuudella. Color correction -ominaisuus mahdollistaa valmiiksi käännetyn pelitason värikorjailun pelin sisällä. Värikorjailun avulla voidaan muokata kentän väritasapainoa tavoitellun tunnelman aikaansaamiseksi (kuvio 15). (Valve Developer Community 2013e.) Riviera-tasossa käytin värikorjailua tuomaan jo entuudestaan puhtaita päävärejä enemmän esille nostamalla värikylläisyyttä suuremmaksi.

Kuvio 15: Color Correction Source Engine -pelimoottorilla

Äänimaiseman loin helppokäyttöisellä Soundscape-ominaisuudella, jolla voidaan rikastaa kentän äänimaisemaa toistuvilla sekä satunnaisesti ilmestyvillä ääniefekteillä. Tasoon sijoitettavat env_soundscape-entiteetit sisältävät tarvittavan informaation aktivoidusta äänimaisemasta sekä tämän vaikutusalueesta. (Valve Developer Community 2011c.) Välimerellisen rantakaupungin oleellisimmat äänet olivat luonnollisesti meren humina, lokkien laulu ja yleisesti liikenteen hälinä. Äänimailmaa syvensin erilaisilla pienemmän vaikutusalueen efekteillä asettamalla kenttään ambient_generic-entiteettejä, jotka sisälsivät muun muassa musiikkia ja auton varashälyttimen äänen.

4.6.2 Optimointi

Suurin osa Source-pelimoottorille rakennettavasta pelitason optimoinnista tapahtuu jo suunnitteluvaiheessa, sillä tärkein optimoinnin osa-alue on järkevästi rakennettu navigaatiorakenne, jossa ruudulle piirrettävät pelialueet pysyvät riittävän hillittyinä ja kevyinä prosessoida. Kenttäsuunnittelun aiemmassa tuotantovaiheessa on myös useita tilanteita, joissa voidaan optimoida suorituskykyä jossain määrin. Näitä tilanteita ovat esimerkiksi detail- ja displacement-brushien oikeanlainen käyttö. Viimeistelyvaiheessa voidaan kuitenkin tehdä vielä huomattavia parannuksia tason tekniseen suorituskykyyn. (Valve Developer Community 2011a; MangyCarface 2009; Van Hoorn, Ralph.)

Source-pelimoottorille rakennettu taso täytyy olla täysin suljettu sitä ympäröivästä avaruudesta. Pelitason kääntämisprosessissa VVIS-komentolinja laskee tarvittavat piirtoalueet ja tallentaa ne BSP-tiedostoon parantaen kentän tehokkuutta. Mikäli tasossa on aukko sitä ympäröivään avaruuteen ei kyseistä VVIS-komentolinjaa voida suorittaa. Tällöin pelikentästä tulee todella raskas prosessoida. (Valve Developer Community 2011a.)

Toinen huomattava optimointikeino on vaikuttaa VVIS-komentolinjan laskemiin piirtoalueisiin niin kutsutuilla hint-brusheilla. Hint-brushit sijoitetaan pelikenttään samalla tavalla, kuin mitkä tahansa muutkin rakenteet. Näiden brushien avulla käyttäjä voi itse kontrolloida milloin pelimoottori piirtää mitäkin piirtoalueita. (Valve Developer Community 2011a; MangyCarface 2009; Van Hoorn, Ralph.)

Muita vähäpätöisempiä optimointikeinoja ovat muun muassa tason maksimipiirtoetäisyyksien asettaminen 3D-malleille ja muulle geometrialle sekä nodraw-materiaalin käyttö, joka pinnalle asetettuna ei piirry lainkaan pelissä vähentäen piirrettävien pintojen määrää. Myös fyysisiä ominaisuuksia sisältävien 3D-mallien ja dynaamisten valojen käytön hillitseminen voi keventää laskentataakkaa. (Valve Developer Community 2011a; MangyCarface 2009; Van Hoorn, Ralph.)

5 Yhteenveto

Opinnäytetyöni tavoitteena oli selvittää ja ratkaista keskeisimmät ongelmat ja haasteet tasosuunnitteluprosessissa Source Engine -pelimoottorilla. Aiempi kokemus käytetyistä ohjelmista ja työkaluista helpotti tuotantovaihetta huomattavasti. Suurin mielenkiintoni tutkielmassa kohdistui pelitason loppuun saattamisessa aina julkaisuvalmiiksi tuotteeksi asti, sillä aiemmat tasosuunnitteluprojektini ovat jääneet kesken uusien ideoiden myötä. Houkutus siirtyä uuden pelitason rakentamiseen oli aina ollut liian suuri.

Avainasemassa onnistumiseen oli entistä kattavampi ideointi ja suunnitelma aloitettavasta työstä. Lisäksi keskittyminen prosessin eri vaiheisiin yksi osa-alue kerrallaan edisti huomattavasti mielenkiintoni säilymistä loppuun asti. Palautteen saaminen ja etenkin sen vastaanottaminen muilta alan osaajilta oli korvaamaton apu.

Mielestäni tämän opinnäytetyön myötä valmistunutta pelikenttää voidaan pitää onnistuneena. Riviera-tason varaan rakennettu portfolio auttoi allekirjoittanutta työllistymään pelialalle välittömästi julkaisun jälkeen. Valmistuneessa projektissa kulminoitui niin koulussa kuin itseopiskelunkin myötä opitut teoriat ja toimintatavat. Työnäytteessä pyrin osoittamaan tietotaitoni työkalujen hallinnassa, graafisessa suunnittelussa, taiteessa, arkkitehtuurissa ja pelimaailmaan liittyvässä psykologiassa.

Pelimekaanisten ominaisuuksien suunnitteluun ja esityöhön sekä visuaalisen ilmeen hiomiseen jäi edelleen parantamisen varaa. Projektin aikana vastaan tuli useita ongelmakohtia, joita en osannut odottaa suunniteluvaiheessa. Tasosuunnittelu on kuitenkin jatkuvaa oppimista, joten kehittyminen ja ongelmatilanteiden kohtaaminen jokaisen tuotantoprosessin aikana ja jälkeen on luonnollista ja jopa toivottavaa.

Lähteet

Hourences 10.12.2006. Lighting in game environments - The Hows and Whys. Mod DB. [verkkodokumentti] <http://www.moddb.com/tutorials/lighting-in-game-environments-the-hows-and-whys> (luettu 22.5.2013).

Jenssen, Magnar 2012. Functional Lighting. [verkkodokumentti] <http://magnarj.net/article_funclight.html> (luettu 22.5.2013).

Johnston, David 1/2003. What is Level Design? [verkkodokumentti] <http://www.johnsto.co.uk/design/level_design> (luettu 22.5.2013).

Lappalainen, Elina 17.4.2013. Villeimmätkin huhut alimitoitettuja - Supercell tekee päivässä 1,8 miljoonaa euroa. Talouselämä. [verkkodokumentti] <http://www.talouselama.fi/uutiset/villeimmatkin+huhut+alimitoitettuja+ +supercell+tekee+paivassa+18+miljoonaa+euroa/a2180327> (luettu 22.5.2013).

MangyCarface 13.12.2009. Optimization in Source: A Practical Demonstration. [verkkodokumentti] <http://www.nodraw.net/2009/12/optimization-in-source-a-practical-demonstration/> (luettu 22.5.2013).

Schwaber, Ken & Sutherland, Jeff 10/2011. The Scrum Guide. Scrum.org. [verkkodokumentti] <www.scrum.org/Portals/0/Documents/Scrum %20Guides/Scrum_Guide.pdf> (luettu 22.5.2013).

Turtola, Ilona 3.4.2013. Rovion pelejä ladattu 1,7 miljardia kertaa - odottaa kasvun jatkuvan. Yle Uutiset. [verkkodokumentti] <http://yle.fi/uutiset/rovion_peleja_ladattu_17_miljardia_kertaa_-_odottaa_kasvun_jatkuvan/6563009> (luettu 22.5.2013).

Twister Homepage 2009. What Is Twister? [verkkodokumentti] <http://www.users.on.net/~imperialcarp/Twister/Twister.html> (luettu 22.5.2013).

Valve Developer Community 2011a. Optimization (level design). [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Optimization_%28level_design%29> (luettu 22.5.2013).

Valve Developer Community 2011b. Material. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Material> (luettu 22.5.2013).

Valve Developer Community 2011c. Sound and Music. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Sound> (luettu 22.5.2013).

Valve Developer Community 2012a. Source Engine Features. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Source_Engine_Features> (luettu 22.5.2013).

Valve Developer Community 2012b. Entity. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Entity> (luettu 22.5.2013).

Valve Developer Community 2012c. Displacement. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Displacement> (luettu 22.5.2013).

Valve Developer Community 2012d. Skybox Basics. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Skybox> (luettu 22.5.2013).

Valve Developer Community 2013a. Source BSP File Format. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Source_BSP_File_Format> (luettu 22.5.2013).

Valve Developer Community 2013b. Hammer Face Edit Dialog. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Hammer_Face_Edit_Dialog> (luettu 22.5.2013).

Valve Developer Community 2013c. Lighting. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Lighting> (luettu 22.5.2013).

Valve Developer Community 2013d. Cubemaps. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Cubemaps> (luettu 22.5.2013).

Valve Developer Community 2013e. Color Correction. [verkkodokumentti] <https://developer.valvesoftware.com/wiki/Color_correction> (luettu 22.5.2013).

Van Hoorn, Ralph. Half-Life 2 Map Editing Optimization Guide. [verkkodokumentti] <http://rvanhoorn.ruhosting.nl/optimization.php?chapter=intro> (luettu 22.5.2013).

Wikipedia 2013a. Video game development. [verkkodokumentti] <https://en.wikipedia.org/wiki/Video_game_development> (luettu 22.5.2013).

Wikipedia 2013b. Scrum (development). [verkkodokumentti] <https://en.wikipedia.org/wiki/Scrum_%28development%29> (luettu 22.5.2013).

Wikipedia 2013c. Source. [verkkodokumentti] <http://fi.wikipedia.org/wiki/Source> (luettu 22.5.2013).

Wikipedia 2013d. Half-Life 2. [verkkodokumentti] <http://fi.wikipedia.org/wiki/Half-Life_2> (luettu 22.5.2013).

Wikipedia 2013e. Counter-Strike: Source. [verkkodokumentti] <http://fi.wikipedia.org/wiki/Counter-Strike:_Source> (luettu 22.5.2013).

Wikipedia 2013f. Source SDK. [verkkodokumentti] <http://en.wikipedia.org/wiki/Source_SDK> (luettu 22.5.2013).

Wikipedia 2013g. Värioppi. [verkkodokumentti] <http://fi.wikipedia.org/wiki/V%C3%A4rioppi> (luettu 22.5.2013).

World Of Level Design 2011a. Why I Failed for Years at Level Design and Game Environments. [verkkodokumentti] <http://www.worldofleveldesign.com/categories/level_design_tutorials/why-i-failed-for-years-at-level-design-and-game-environments.php> (luettu 22.5.2013).

World Of Level Design 2011b. How to Plan Level Designs and Game Environments. [verkkodokumentti] <http://www.worldofleveldesign.com/categories/level_design_tutorials/how-to-plan-level-designs-game-environments-workflow.php> (luettu 22.5.2013).

Kuvankaappaukset valmiista Riviera-pelikentästä